Nationaal Lucht- en Ruimtevaartlaboratorium

National Aerospace Laboratory NLR

Executive summary

Distributed Debriefing in Collective Mission Simulation

Problem area

Mission training and rehearsal for both real and simulated mission exercises become increasingly more distributed. This has consequences for debriefing, since the participants are at various locations. Therefore it is more difficult to establish a universal understanding of the events of the mission. To overcome this, playback of mission data of every different location should be synchronized and at least one location should control the playback. Furthermore the different operators are all using their own debrief systems that currently aren't interoperable with each other. These different systems are developed for giving the best view of each platform. The operators are used to these, know how to operate the tools and get the best views out of the tools for their own debriefing. It is our purpose to make use of these tools but getting these to that same event may be challenging

Description of work

In the SISO community a Study Group has discussed to develop a standard for distributed debriefing, i.e. the Distributed Debrief Control Protocol (DDCP). The DDCP protocol enables multiple separated recorders / replayers to be controlled (start, stop, pause) in a time-synchronized manner. Since

Report no.
NLR-TP-2010-663

Author(s)
R.T.A. Meiland
A.J.J. Lemmers

Report classification
UNCLASSIFIED

Date
July 2011

Knowledge area(s)
Training, Simulatie en Operator Performance

Descriptor(s)
Collective Mission Simulation
Distributed debriefing
Distributed Debrief Control Protocol
Supporting systems

This report is based on a presentation held at the SISO Simulation Interoperability Workshop, Orlando (FL), U.S.A., 20-24 September 2010.

the debrief tool Personal Computer Debrief System (PCDS) is used by the Royal Netherlands Air Force (RNLAF), we have incorporated PCDS into our DDCP implementation. Furthermore we have implemented a part of the DDCP and are able to handle a selected set of the messages, namely the set for synchronizing the playback of mission data. This way the operators can use the same debrief system they are used to.

A practical approach has been followed to demonstrate the working of distributed debriefing where in a use case the evolving standard Distributed Debriefing Control Protocol (DDCP) is tested in a distributed debriefing set-up with the operational debriefing tool PCDS. In a distributed set-up with DDCP tools at two dispersed locations we demonstrated that the DDCP protocol can support the synchronisation of operational debriefing tools at dislocated sites

Results and conclusions

In a distributed set-up with DDCP tools at two dispersed locations we demonstrated that the DDCP protocol can support the synchronisation of operational debriefing tools at dislocated sites. We were able to incorporate PCDS into the DDCP software implementation for synchronized playback. The first reactions from the fighter pilot community on this debriefing set-up indicate that this provides a useful addition to the distributed debrief. We recommend therefore to continue with the DDCP standardisation effort and to establish a Product Development Group (PDG) for this in the SISO community.

Applicability

The Distributed Debrief Control Protocol could be the solution to enable synchronized playback of mission data at different locations. However the DDCP hasn't matured yet to become a standard and further research and standardisation efforts need to be done

Nationaal Lucht- en Ruimtevaartlaboratorium, National Aerospace Laboratory NLR

Anthony Fokkerweg 2, 1059 CM Amsterdam,
P.O. Box 90502, 1006 BM Amsterdam, The Netherlands
Telephone +31 20 511 31 13, Fax +31 20 511 32 10, Web site: www.nlr.nl

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2010-663

Distributed Debriefing in Collective Mission Simulation

R.T.A. Meiland and A.J.J. Lemmers

This report is based on a presentation held at the at the SISO Simulation Interoperability Workshop, Orlando (FL), U.S.A., 20-24 September. 2010.

This publication has been refereed by the Advisory Committee AIR TRANSPORT.

Customer	Ministry of Defence
Contract number	GBS 06/124
Owner	NLR
Division NLR	Air Transport
Distribution	Unlimited
Classification of title	Unclassified
	July 2011

Approved by:

Author	Reviewer	Managing department
R. Meiland 07-07-'11		7-7-'11

Contents

Distributed Debriefing in Collective Mission Simulation

Remco Meiland, Arjan Lemmers
National Aerospace Laboratory NLR
Anthony Fokkerweg 2
1059 CM AMSTERDAM
The Netherlands
Meiland@nlr.nl, Lemmers@nlr.nl

ABSTRACT: Mission debriefing is a crucial part of mission training and rehearsal and an important source for gaining understanding of one's actual performance. During debriefing participants get feedback about their actions and performance within a larger context, and in relation to other participants. Representation of this feedback relies heavily on tool support that needs to balance between the right information in the right doses at the right time. At the moment, the National Aerospace Laboratory NLR in Amsterdam, The Netherlands, is involved in a NL MoD program called "Collective Mission Simulation" (CMS). Within this program several experiments have been held to test and evaluate various aspects of collective mission simulation. One of these aspects is distributed debriefing. The main purpose of our research is to assess several options for a distributed setup that gives military operators the means to effectively debrief in a collective mission. A special focus of our research is to enable the current operational debriefing tools to be used in this setup. From this point of view we focus on aspects such as how to set up the debriefing environment, how to run the debriefing, what information to share, and how to do this at the collective level.

In the SISO community a Study Group has discussed to develop a standard for distributed debriefing, i.e. the Distributed Debrief Control Protocol (DDCP). During mission execution various data-streams have been recorded at various locations, such as audio communication, video, simulation data, etc. During debriefing these multiple recorded streams are being replayed. The DDCP protocol enables multiple separated recorders / replayers to be controlled (start, stop) in a time-synchronized manner. The CMS program has followed this initiative and set up a test case in which the functionalities and usability of DDCP have been assessed. In this paper we will outline the technical findings of our research including the design and development of our distributed debriefing test bed and the technical implementation of DDCP. These aspects are evaluated in a use case with manned simulators at two distributed locations.

1. Introduction

As in real military operations, military training is becoming more distributed in nature. This has led to a growing need for a collective mission simulation (CMS) environment that can support concept development and experimentation, e.g. in the areas of mission training and rehearsal, system acquisition, tactics and doctrine development, and command and control. CMS provides the military with a distributed simulation environment that allows units to participate from their own base in distributed mission training events.

Mission training and rehearsal rely heavily upon bringing all participating teams, platforms, and command and control, into a relevant, coherent and realistic mission environment. Debriefing is one of the crucial aspects as it is an important source for gaining understanding of actions of both individuals and teams in relation to the mission outcome. During debriefing the participants get feedback about their performance within a larger context, and in relation to the other participants. Representation of this feedback in an appropriate and relevant manner calls for innovative debriefing solutions.

At the moment, the National Aerospace Laboratory NLR in Amsterdam and TNO in The Hague, The Netherlands, are involved in a NL MoD program called “Collective Mission Simulation” (CMS) [1]. As part of this research program NLR has investigated possibilities to set up a distributed debriefing environment. A practical approach has been followed where in a use case the evolving standard Distributed Debriefing Control Protocol (DDCP) is tested in a distributed debriefing set-up with the operational debriefing tool Personal Computer Debrief System (PCDS).

In this paper we will describe this use case and we will shortly explain the basic principles of the DDCP. Then we explain our experiences on implementing (parts of) this protocol. We will conclude with describing the results of the use case in which we tested the debriefing set-up with military operators giving their feedback.

2. Debriefing needs

The main difference between mission training at one location and distributed mission training is obviously the fact that participants are at various locations. Of course this has consequences for debriefing as it is more difficult to establish a universal understanding of the events of the mission. To overcome this, playback of mission data of every different location should be synchronized and at least one location should control the playback. Once the mission playback of all geographically separated sites are synchronized, it allows every team to give input from their point of view or platform position. Having multiple views of the mission represented, you can then de-conflict misunderstandings that may have caused mission degradation [6]. Other ways of

operation or communication can then be discussed to anticipate future misunderstandings.

Let us start with a practical example of an extraction mission. During an exercise 2 teams are involved, one F-16 team and a helicopter team (Eurocopter Cougar AS 532U2). They are both on different places far away from each other. Both teams fulfil different roles. The helicopter team's objective is to pickup someone from an extraction point behind the FLOT (Forward Line Of Troops). The F-16 team has to support the extraction by suppressing OpFor air and ground assets. Communication between both teams is of crucial importance. The F-16 team has to inform the helicopter pilot when it is allowed to cross the FLOT to extract the person. The helicopter pilot informs the F-16 team about the extraction status.

Figure 1 Cougar helicopters

Of course both teams can debrief only internally, but more lessons are learned by distributed debriefing. The teams brief not only themselves but they brief together. That makes the teams much more aware of what their role in the mission outcome is, and how the other teams perceived their actions.

According [4] a distributed debrief occurs when multiple geographically separated parties desire to conduct a debrief as if they are all at the same location. How can this be achieved?

3. Distributed Debriefing Control Protocol as a solution

Currently, the different operators are all using their own debrief systems that aren't interoperable with each other. These different systems are developed for giving the best view of each platform. The operators are used to these, know how to operate the tools and get the best views out of the tools for their own debriefing. It is our purpose to make use of these tools but getting these to that same event may be challenging.

Therefore a need exists for protocols that can bridge the gap between the systems which would allow participants to focus on the debrief and not be distracted by trying to get the systems synced up. A protocol would also facilitate a more interactive exchange of ideas for distributed participants by giving them some of the advantages that collocated participants share. In the SISO (Simulation Interoperability Standards Organization) [2] community the initiative is taken to define a standard protocol for controlling distributed debriefing, the DDCP. In 2007 a study group has been established to evaluate industry and government interest in developing a distributed

debrief control protocol standard. This study group has sent out a survey to record the needs.

In 2008 the results of the survey are published [6]. Unfortunately it seems that the initial momentum of the Study Group has waned and the start of a new Product Development Group (PDG) is delayed until further interest is generated. Nevertheless, the last question of the survey queried the interest in supporting a PDG and 88% of the responses were in favour of supporting a PDG. Ideally the initial standard would focus on the record and playback capabilities, which provide the backbone for the distributed debrief capability.

The capability to control the playback of a simulation is characterized in the following ways:

- Play – The ability to start the playback.
- Pause – The ability to pause the playback.
- Fast Forward – The ability to advance through the playback in predetermined increments.
- Rewind – The ability to advance the playback in reverse through pre-determined increments.
- Seek – The ability to seek to a new playback time in a short or near-instantaneous amount of time.

We consider DDCP as an interesting solution to control and synchronize playback of mission data and multimedia content among training devices across a long-haul network during Mass Distributed Debrief operations. Therefore we support the DDCP initiative and start implementing parts of the DDCP. This will gain us valuable insights into the protocol itself, which we will feed back to the study group and a possible upcoming DDCP PDG to improve the standard.

Our implementation of the Distributed Debriefing Control Protocol (DDCP) is based on the specification of The Boeing Company [3]. The implementation is far from complete, we intended to evaluate the mechanism of synchronized and controlled event replay of data across multiple computers and how distributed debriefing can contribute to improve debriefing in CMS.

A DDCP package contains one packet header and one or more records. The header and records consist of several fields. These records are for example meant for session management such as loading and saving a session, for group management, i.e. joining and leaving a group, or for playback control, such as play, pause, fast forward and rewind the playback.

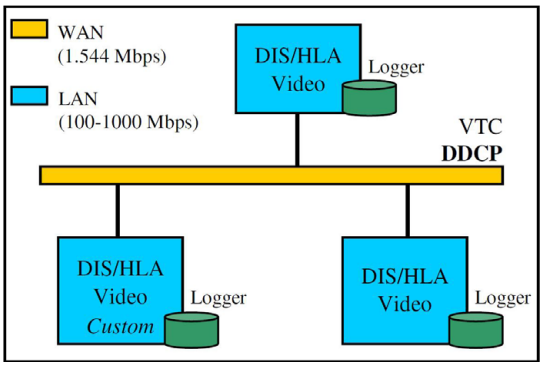

Figure 2: Synchronized Debrief

To limit bandwidth usage and for security reasons, only control messages are sent over the network, not the data itself. A scheme is

represented in Figure 2. It is expected that each site records data independently [4]. Once the data is recorded, each site can playback its own recording in a synchronized manner by using the DDCP protocol.
For sending the control messages over the network, the principle of serialization / deserialization is used. The protocol description is unambiguous regarding the field sizes and types (for instance unsigned / signed 32-bit integer) of the packet and various records. Furthermore byte order (little-endian, big-endian) is reckoned with.

The protocol is implemented in C++ using the User Datagram Protocol (UDP) as the transport layer, since DDCP is designed to operate directly on top of UDP. We can easily exchange data through the mechanism of serialization / deserialization and deal with Endianness.

Design patterns [5] are chosen as a software design strategy for the implementation of the DDCP protocol in C++. We have used a factory pattern to create the packet and records as described in [3]. Furthermore this factory pattern is used to determine the byte order of the records.

Although we can create, send and receive all records, our implementation of handling the records is limited. Since the record and playback capabilities provide the backbone for the distributed debrief capability, we focussed on handling the records that are needed to control playback.

The ability to control the logger depends if the tool features some sort of API allowing for the user to produce and then integrate new record, play, stop, and seek capabilities to the existing system.

We have several loggers at our disposal, i.e. MÄK Data Logger and an internal logger in the Personal Computer Debrief System (PCDS) which is shown in Figure 3. A C++ API and Interface Control Document are received for PCDS for record / replay control. MÄK Data Logger contains a C++ API. Direct control via the API of the considered Loggers, motivates our choice for implementing DDCP in the C++ programming language.

The MÄK API has very extensive control prospects compared to the PCDS API. On the other hand since the Royal Netherlands Air Force (RNLAF) uses PCDS, it is interesting to evaluate integrating the DDCP protocol with the PCDS API. Also mission replay can be directly viewed by PCDS and not by MÄK Data Logger.

The DDCP specification [3] describes a master – slave communication model where one device has unidirectional control over one or more other devices.

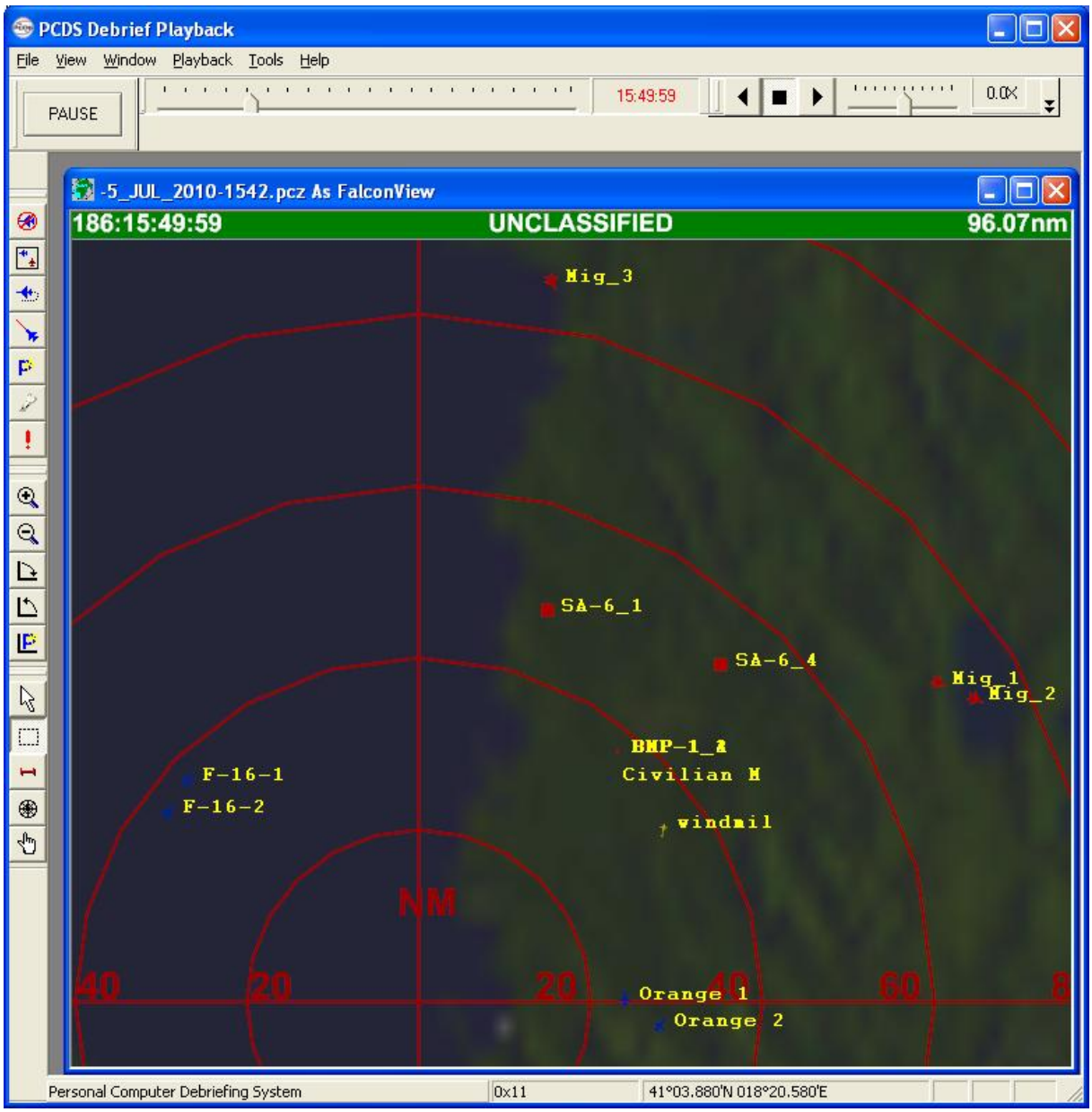

Figure 3 Personal Computer Debrief System

In our implementation, for the master device it means that when a logger control event happens (pressing play, stop, pause, dragging the time slider, etc) a SYNC Notification Record has to be created with the correct values of the state, effective time, requested time, etc. and send the package to the remote loggers using UDP. The slave device commands the logger to play, stop, seek, etc. when a SYNC Notification record is received.

Figure 4 depicts the state diagram for client devices. A client device starts in the ALONE state and enters the STOP state when it receives a JOIN request [3].

A GUI (Graphical User Interface) is designed to control DDCP network traffic and logger playback. The framework we have chosen is Qt by Trolltech which is based on C++, is open source and provides us with all the functionality needed to build complex, high-performance GUI and console applications. Furthermore Qt provides single-source portability across Microsoft Windows, Mac OS X, Linux, all major commercial Unix variants, and embedded Linux.

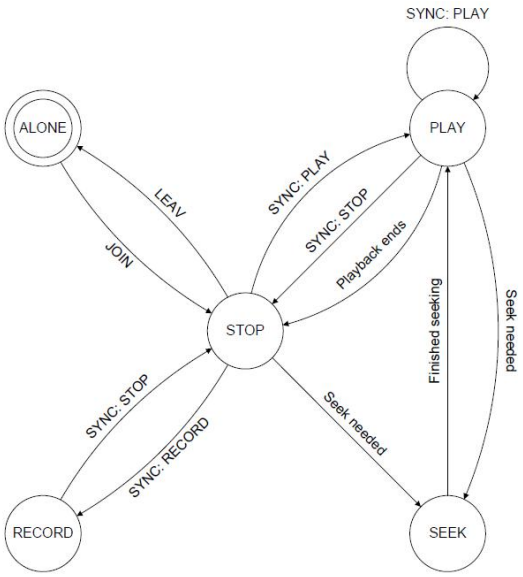

Figure 4 State Diagram

The software scheme in Figure 5 depicts the software implementation. Connected blocks interact with each other. An arrow denotes a 'has a' relationship. For instance the packet handler has a Sync Notification Record but the Sync Notification Record doesn't know anything about the Packet handler, since the arrow points only in one direction. Several times in the scheme of Figure 5 the arrow points both ways, for instance the Mediator has a logger interface, but the logger interface knows about the mediator as well. Actions go both ways, so the mediator can influence the logger interface by giving a play command, but the logger interface can also tell the mediator it has stopped playback, for instance. Finally a dot represents an 'is a' relationship, so a MÄK logger is an implementation of the logger interface <ILogger>.

Serveral software blocks will be explained next:

- UDPSocket: communicates with the outside world and deals with serialization / deserialization
- Graphical User Interface: to configure the UDP Socket ports, subnet mask, show received data, select master or slave, etc.
- Mediator: connects and controls most classes. The separate classes are only aware of the mediator and are decoupled from other classes.
- State Manager: manages the current state, which can be ALONE, STOP, PLAY, RECORD and SEEK. It also tracks if a transition to another state can be done, for example in ALONE we can only go to STOP and not PLAY. State transitions are based on Figure 4 State Diagram.
- Package Handler: handles a received DDCP packet (in case of a slave device) or it creates a packet to be sent (in case of a master device).
- ILogger: logger interface. Playback control is communicated to the logger interface and not to implementations of the logger interface. This way,

extensions of new logger implementations can be easily made.

- MÄK / PCDS Logger: are our concrete logger implementations, which can play, stop, record and seek DIS data. Future implementations could be a video logger for instance showing on-board cockpit video.

Dashed blocks represent not yet implemented software blocks, so for instance only records for playback control (Sync Notification) can be handled at the moment.

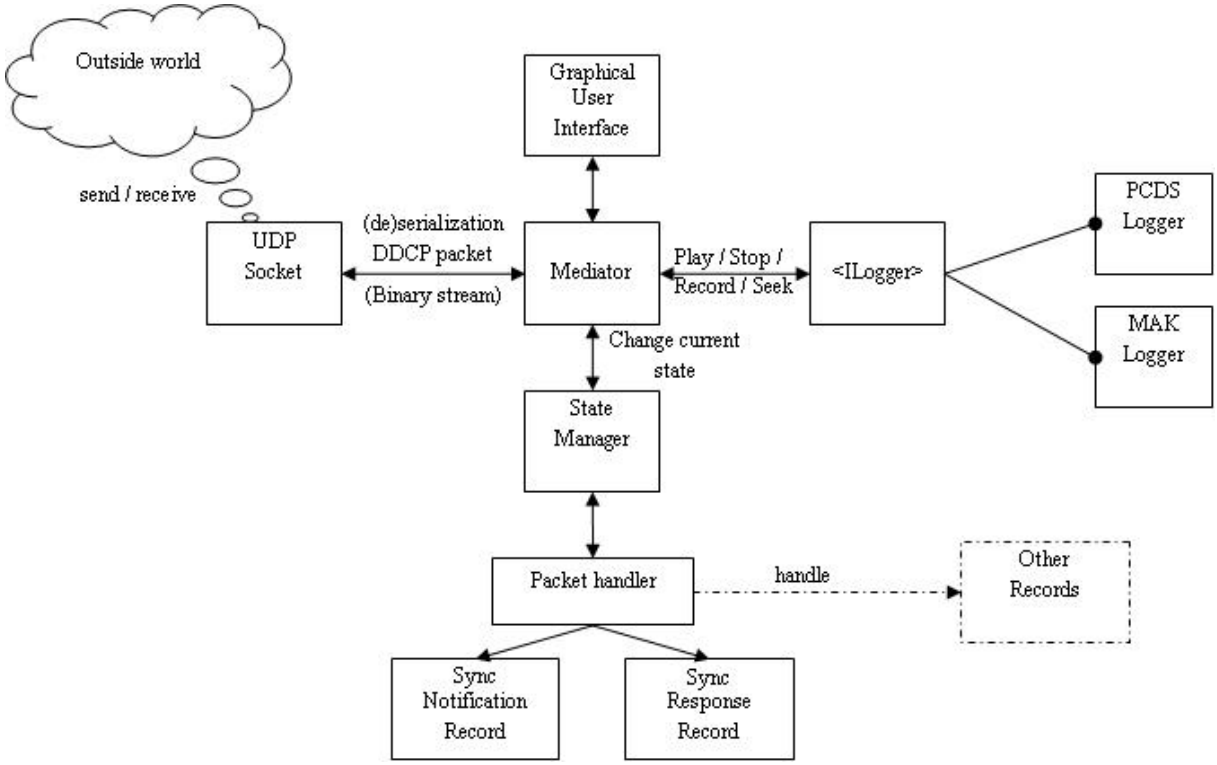

Figure 5 DDCP Software Scheme

Both the PCDS Logger and the MÄK Logger are implementations of the interface <ILogger> as is shown in Figure 5. This logger interface has the following commands:

- stop – stops the recording or playback.
- play – plays the logger at a certain playback speed.
- record – records a future playback.
- seek – seeks to a certain playback time.

Because of this software design, it's easy to upgrade the application with a new logger, for instance on-board cockpit video. Only the new logger has to be implemented, the rest of the code blocks remain unchanged. Important playback speed control options we described earlier, such as pause, fast forward or rewind are incorporated in the play command. The play command has an argument to control the playback speed. This way besides normal

playback, the logger can also pause, fast forward or rewind the playback by changing the playback speed.

Although the DDCP packet description is very much unambiguous, we believe handling DDCP packets will not be naturally the same between independently created implementations of the protocol. The interaction between master and slave is a key factor of successful replay between several dislocated replayers. Therefore further research is needed and we intend to report our findings in future work. However this version of the DDCP protocol is already a great step forward to distributed debriefing.

4. Use case

The use case is held amongst two sites, and only consists of air forces, namely one F-16 team and a helicopter team (Eurocopter Cougar AS 532U2). During a distributed debriefing every site can regard the mission replay from its own perspective. This makes distributed debriefing especially interesting for joint exercises, where the cooperation between e.g. air force, navy and army is practiced, since these perspectives can be very different. For two air sites the perspectives are less different, however distributed debriefing is still interesting. An extraction scenario (Figure 6) is defined where military operational relevance for air forces is considered.

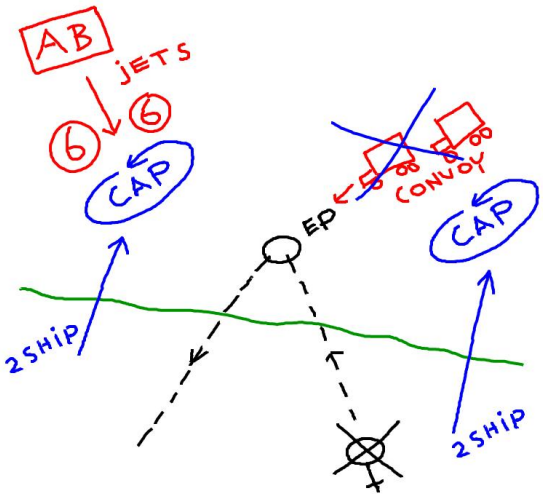

Figure 6 Extraction scheme

A helicopter has to extract a VIP from the extraction point (EP) behind the FLOT. The helicopter is protected by two 2-ship F-16s, performing Combat Air Patrol (CAP) east and west of the EP. One of the 2-ships will be tasked to attack an enemy convoy heading towards the EP.

During the mission execution 2 recordings will be made; one at the F-16 site and one at the Helicopter site. Both sites are located at the National Aerospace Laboratory in Amsterdam, the Netherlands. They are both in different buildings and have separate networks and can therefore be regarded as separate sites, the set-up is shown in Figure 7.

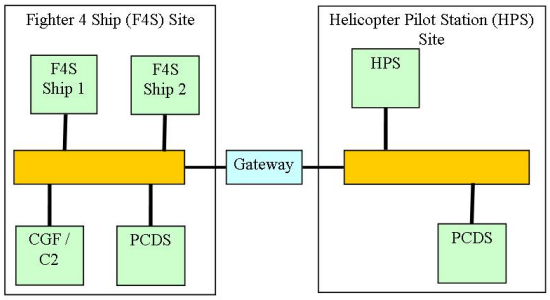

Figure 7 Sites set-up

The Fighter 4 Ship (F4S) is a research simulation facility that can simulate the collective tactical operations of up to four fast-jet fighter aircraft. A 2-ship is used for this use case. One of the ships is depicted in Figure 8. The Helicopter Pilot Station (HPS) is a fixed-base reconfigurable simulator and has been developed with particular emphasis on handling qualities and human factors research.

For playback during debrief each site uses its own recoding only. Synchronised playback of these 2 recordings is used and through Smartboard desktop sharing the sites can show events on each others playback tool to the other.

We have implemented the DDCP protocol on both PCDS and the MÄK Data Logger. This way we have tested that two different playback systems can control and synchronize playback of each other. However during the use case we have only used PCDS for both the F-16 team and the helicopter team. This is because MÄK Data Logger is not used as a debriefing system; it is used as a DIS / HLA logger.

After mission execution, both teams conduct a flight / formation debrief. This means both teams reconstruct the mission from their own perspective, no information is shared between the helicopter team and F-16 team.

Therefore during the flight / formation debrief, lessons learned are identified, mission snapshots are made and material is prepared for the mass debrief.

During the mass debrief both teams should be able to replay the mission for mission reconstruction. Since debriefing deals with mission execution, playback time is an important parameter. For distributed debriefing it is undesirable that participants from different sites are looking at different mission situations because of a non corresponding simulation time. Therefore synchronisation of the playback data of the mission is desired. Manually synchronizing mission playback data isn't very accurate and time effective. It demands two or more instructors to communicate for instance by phone and subsequently seek manually to play the desired mission situation.

Figure 8 F4S Ship1

This set-up has been implemented and tested with three (former) F-16 and Chinook pilots in the loop and their reactions were positive on the value of the distributed debriefing set-up. Unfortunately due to busy schedules of the pilots, the use case experiment itself has been delayed and we can not feedback the experiences of the active fighter and helicopter

pilots. However we do expect a positive opinion of the DDCP protocol. [1]

5. Conclusions and recommendations

As part of the CMS research program NLR has investigated possibilities to set up a distributed debriefing environment. A practical approach has been followed where in a use case the evolving standard DDCP has been tested in a distributed debriefing set-up with the operational debriefing tool PCDS.

We have implemented a selected set of DDCP messages and incorporated these in PCDS for synchronized playback. In a distributed set-up with DDCP tools at two dispersed locations we demonstrated that the DDCP protocol can support the synchronisation of operational debriefing tools at dislocated sites. The first reactions from the fighter pilot community on this debriefing set-up indicate that this provides a useful addition to the distributed debrief. We recommend therefore to go further with the DDCP standardisation effort and to establish a PDG for this in the SISO community.

6. Future work

We will continue to explore the use of DDCP further. As such, our research will continue to contribute to the DDCP SISO standardization initiative, extending the practical implications for this standard, including actual implementation recommendations based upon our findings.

A next major step that we foresee is to expand the CMS debriefing concepts to other collective exercises. That will be the major proof if distributed debriefing is beneficial for the military operators performing their training in these exercises.

References

[1]. Jeroen Voogd, Klaas-jan de Kraker, Lesley Jacobs, Frido Kuijper, Michel Keuning, Rombout Karelse (December 2008), Collective Mission Simulation in The Netherlands, Key Problems & Solutions, *Interservice/Industry Training, simulation and Education Conference (I/ITSEC)*, Orlando

[2]. http://www.sisostds.org/

[3]. Curtis A. Armstrong (December 2007), Specification for Distributed Debrief Control Protocol, The Boeing Company

[4]. Randy Pitz, Curtis Armstrong (November 2007), Advanced Distributed Debrief for Joint and Coalition Training, *Interservice/Industry Training Simulation, and Education Conference (I/ITSEC)*, Orlando

[5]. Gamma, Erich; Richard Helm, Ralph Johnson, and John Vlissides (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2.

[1] The fighter pilot feedback is described in [7].

[6]. Final Report for: The Distributed Debrief Control Protocol Study Group, SISO-REF-028-2008

[7]. A.J.J. Lemmers and R.T.A. Meiland (2010), Supporting Systems for Collective Mission Simulation, NLR-CR-2010-653

Author biographies

Remco Meiland holds an MSc in Electrical Engineering from the Delft University of Technology, with a specialization in control engineering. Afterwards he worked in the field of Information and Communications Technology for 5 years. Since 2009 he works at the National Aerospace Laboratory at the Training & Simulation department as a simulation engineer.

Arjan Lemmers graduated from the Delft University of Technology, where he studied Aeronautical Engineering. Afterwards he joined the Royal Netherlands Navy for 2 years as a lecturer and research fellow for control theory. Since 1998 Arjan is working for the National Aerospace Laboratory NLR in the field of flight simulation, distributed simulation and training technology. Currently he holds a position in the Training & Simulation department as senior R&D manager modeling & simulation. He is the NLR program manager for Collective Mission Simulation. At the same time he is chairman of NATO RTO task group MSG-071 Missionland.